# 模型及代码下载

5-5.0

<https://github.com/ultralytics/yolov5>

<https://github.com/ultralytics/yolov5/releases>

5-4.0

<https://github.com/ultralytics/yolov5/releases/tag/v4.0>

YOLOv5的详细使用教程

<https://blog.csdn.net/weixin_41010198/article/details/106785253>

# 原生代码跑测

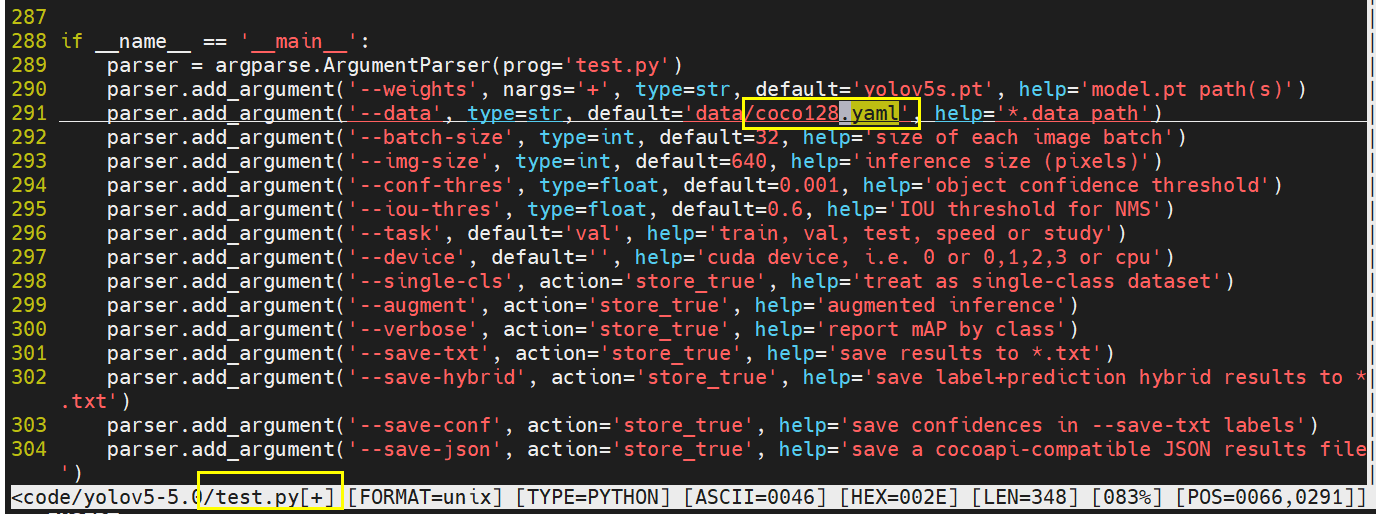
python3 detect.py

python3 test.py --save-json --weights yolov5\*.pt

选择save-json选项才会去做精度比对

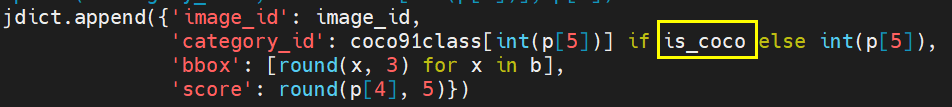
## 用coco128子集跑测

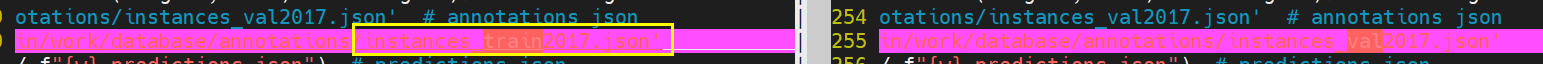
注意如下修改：



1. 

修改此处的coco.yaml为coco128.yaml，否则，不会做coco80~91类的类型映射，但是coco2017的训练及验证集都是基于91类。此处的is\_coco flag会影响后面生成跑测json文件中对应的category\_id。

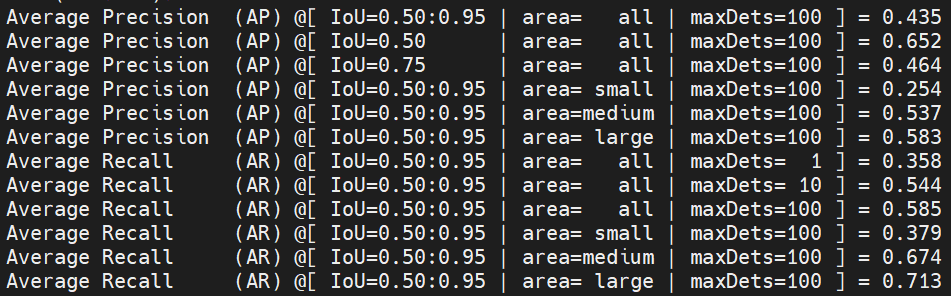


1. 

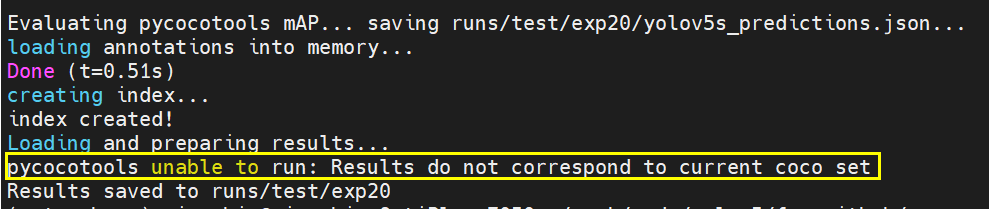
coco128为train2017的子集，因此比对的json文件应该使用annotations/**instances\_train2017.json**

修改正确之后的跑测结果：

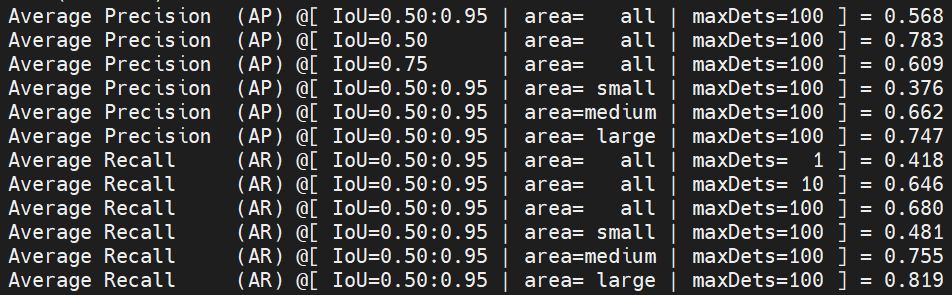
Yolov5s



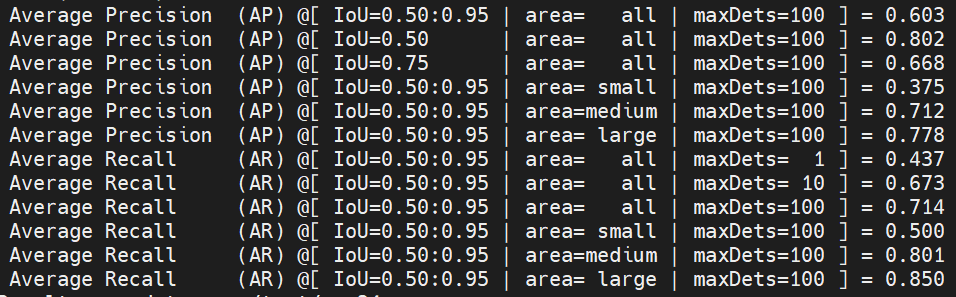
否则会提示：



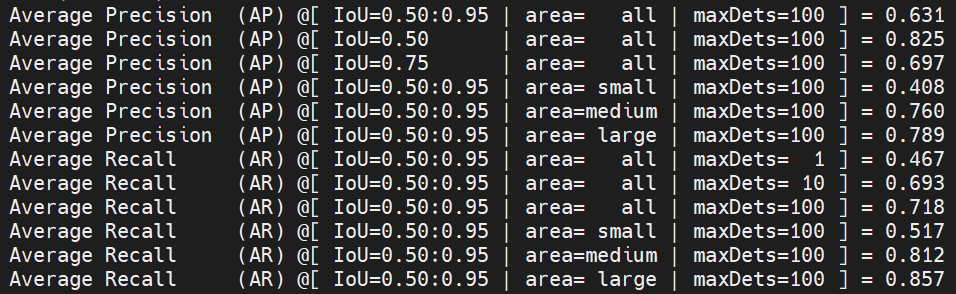
Yolov5m



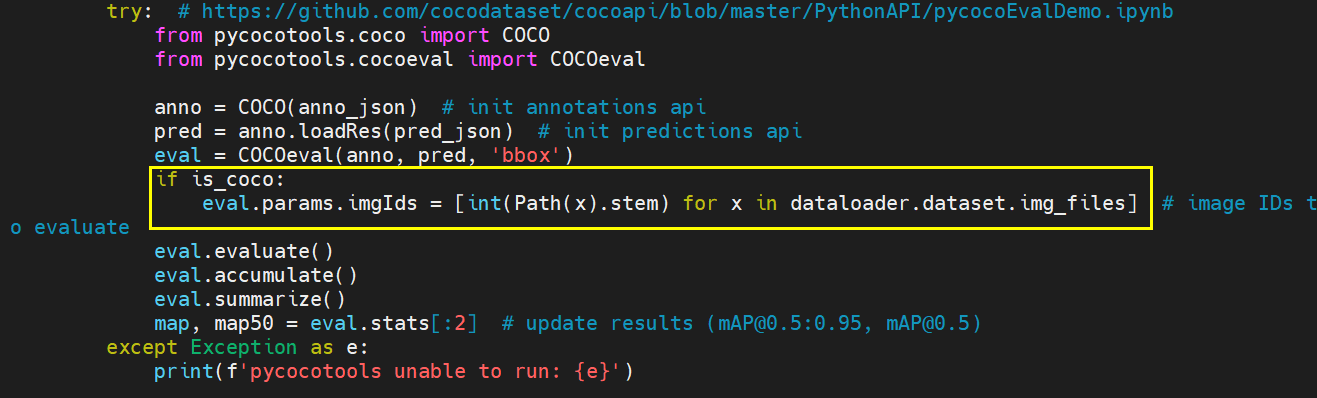
Yolov5l



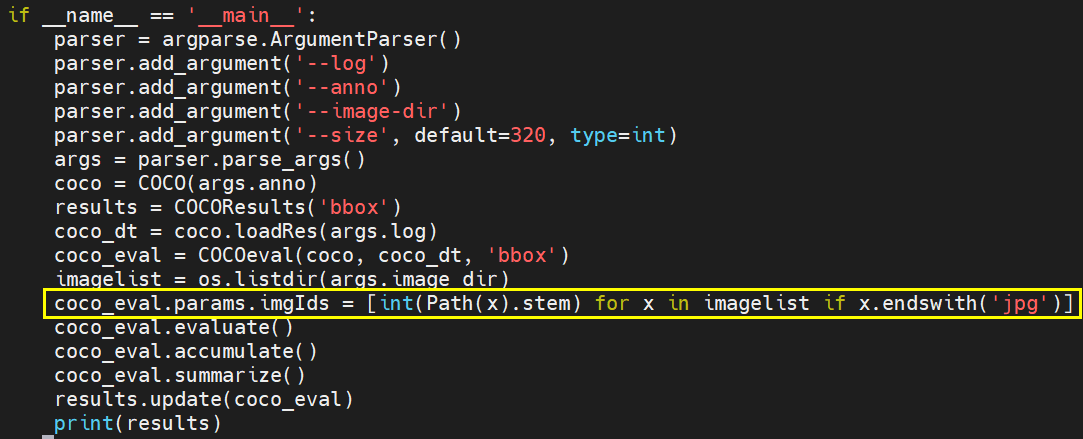
Yolov5x



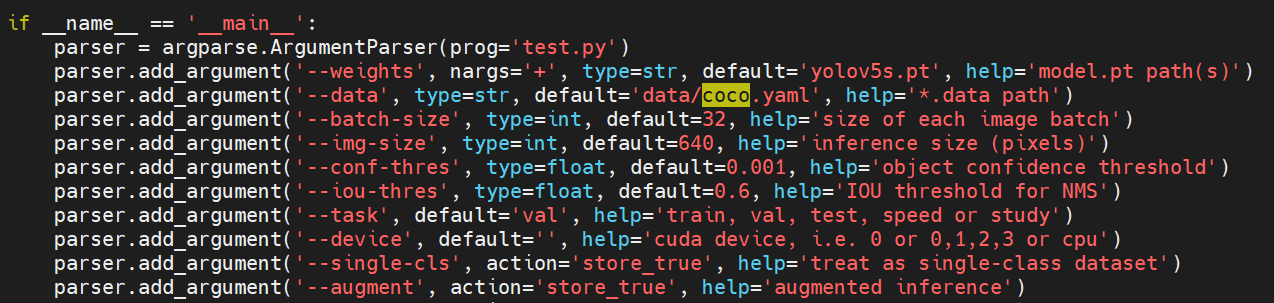
1. 因为是用子集跑测，因此需要在coco eval中传入图片ID的参数，否则计算出来精度非常差



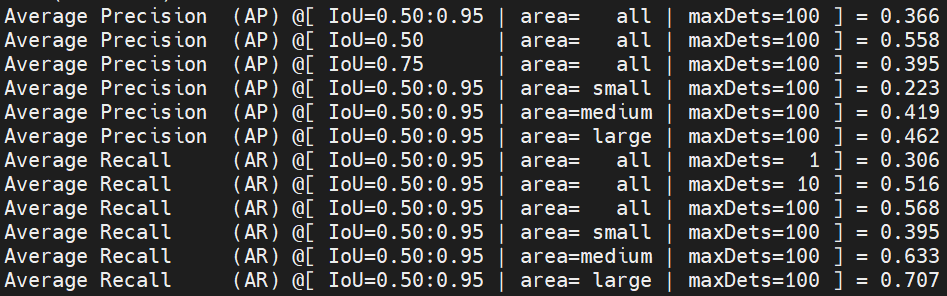
或者：



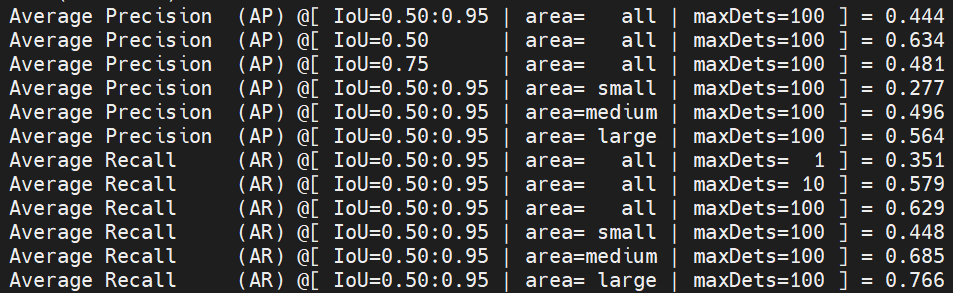
## 用coco跑测



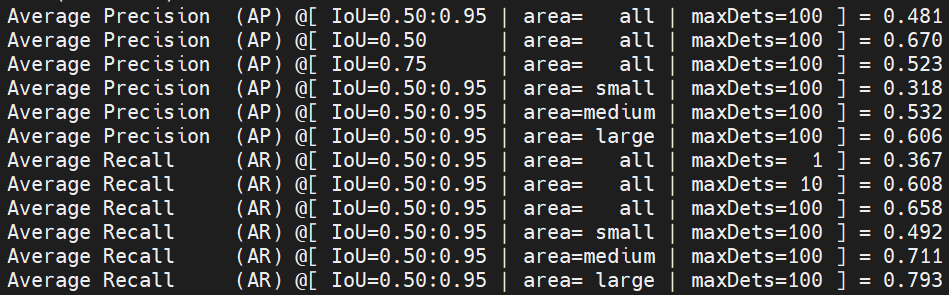
Yolov5s



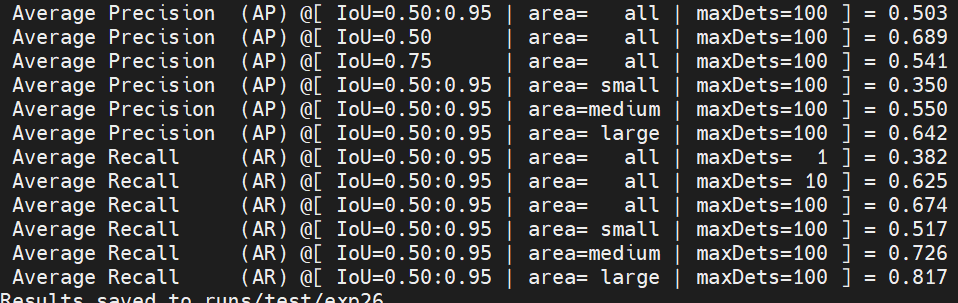
Yolov5m



Yolov5l



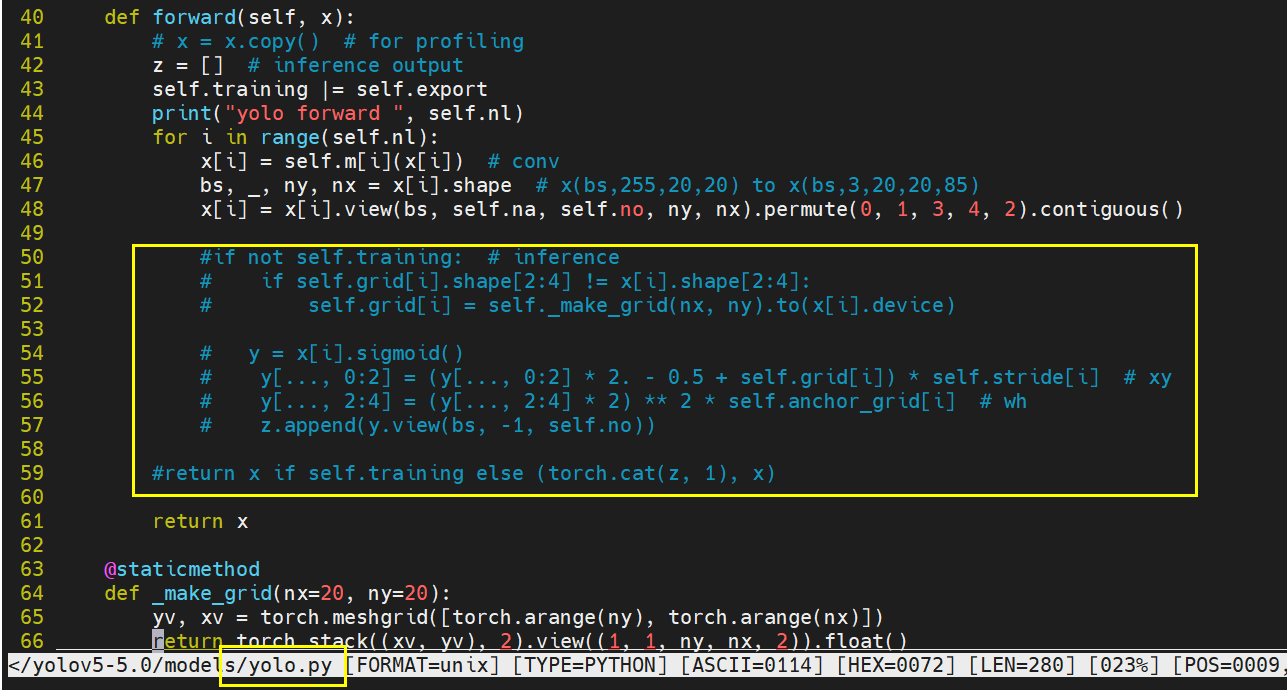
Yolov5x

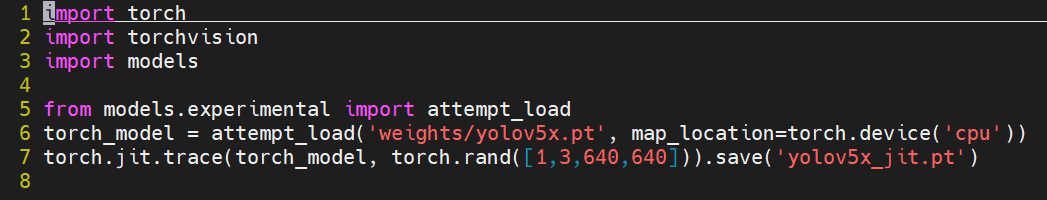


# 模型编译及测试

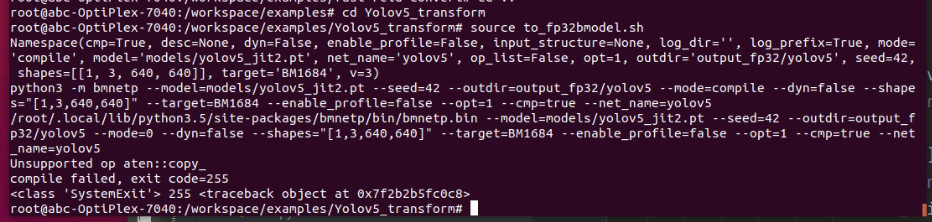
## 模型trace

<https://blog.csdn.net/santanan/article/details/113686650>





否则会报错或者无法成功trace出模型



## bmodel回归测试（coco val2017数据集）

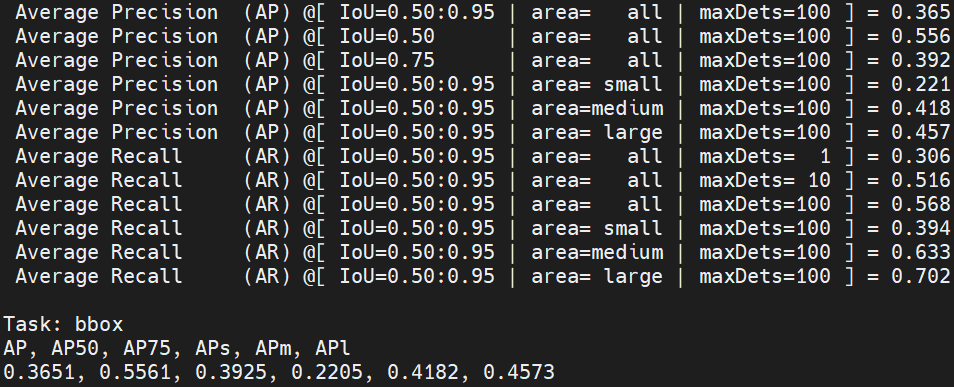
使用sail接口

./regression.sh \*\*\*\*.bmodel \*\*\*\*.json

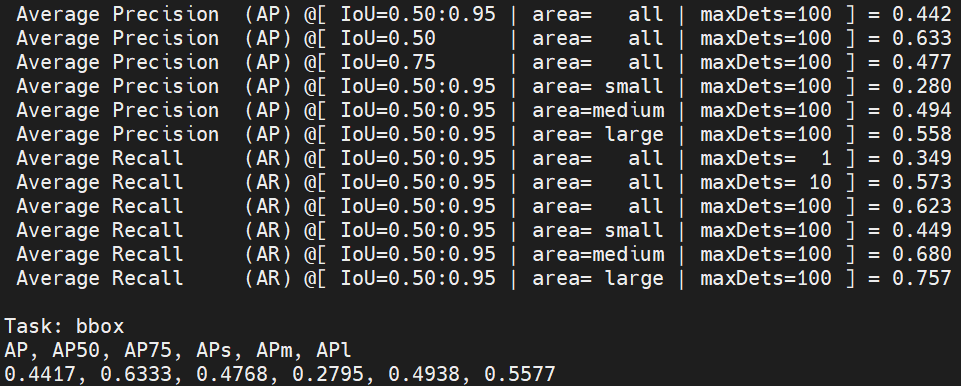
### Fp32 bmodel

python3 -m bmnetp --target=BM1684 --opt=1 --cmp=true --shapes="[1,3,640,640]" --model=path/to/yolov5\*.pth --outdir=output --dyn=false

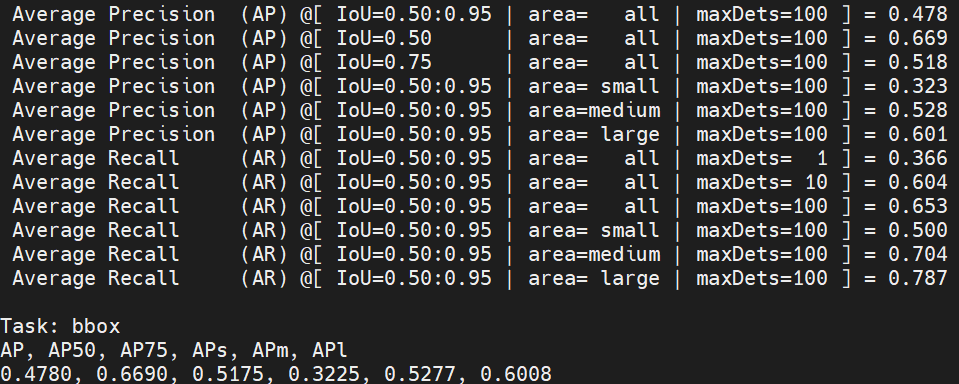
Yolov5s\_fp32.bmodel



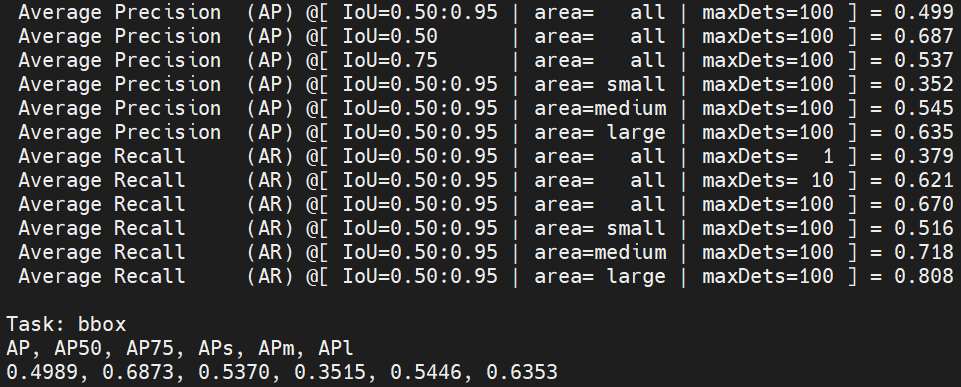
Yolov5m\_fp32.bmodel



Yolov5l\_fp32.bmodel



Yolov5x\_fp32.bmodel



## 量化

### lmdb制作

cd coco/images/val2017

find –name \*.jpg > imagelist.txt

convert\_imageset coco/images/val2017 coco/images/val2017/imagelist.txt --resize\_width 640 --resize\_height 640 /data/yolov5\_640640\_lmdb

### 模型转换

source envsetup\_multi.sh

1. \*.pt to fp32umodel

convert\_yolov5\_pytorchnet ./ yolov5\*

1. 量化

quant\_yolov5\_net yolov5\*

1. 跑测

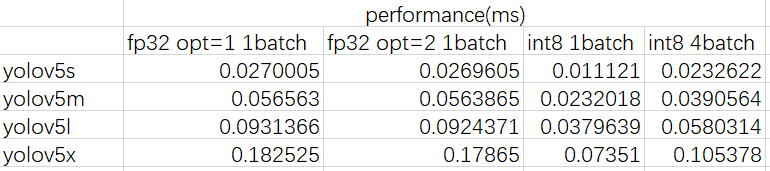
run\_yolov5\_test\_int8 yolov5\*

1. 转bmodel

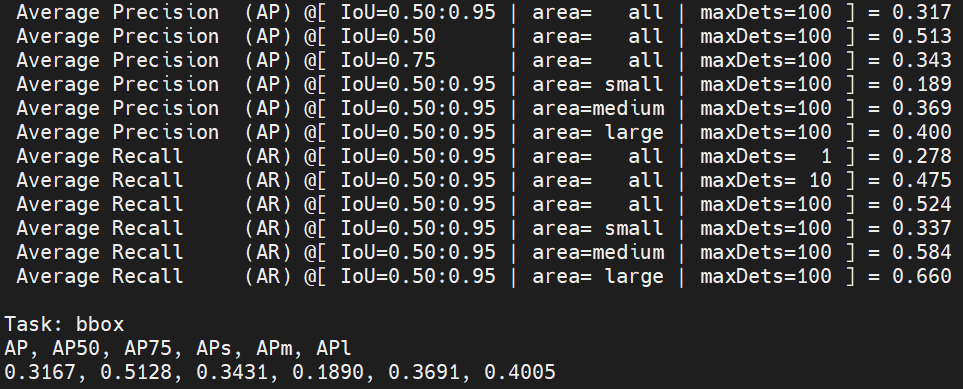
convert\_yolov5\_to\_bmodel yolov5\*

### 量化调优

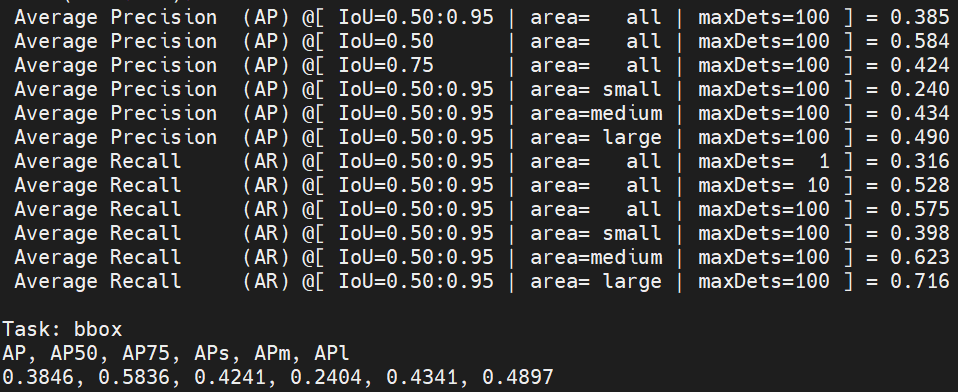
经过多次测试，最终选择-fpfwd\_outputs =’< 24 >14,< 24 >51,< 24 >82’。



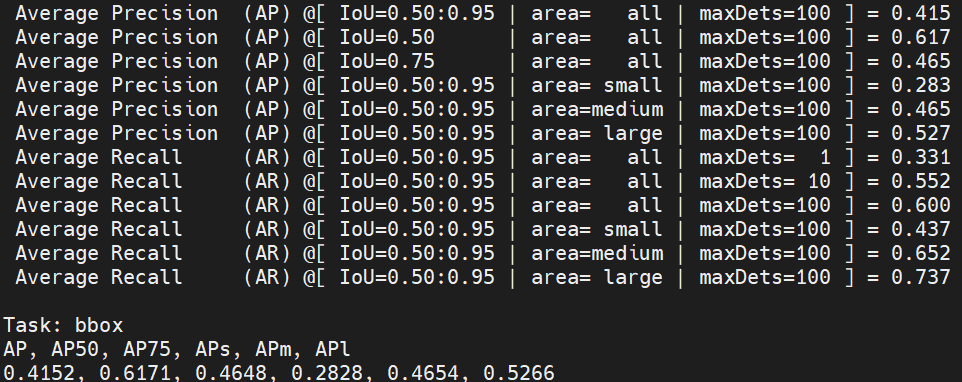
Yolov5s-int8.bmodel



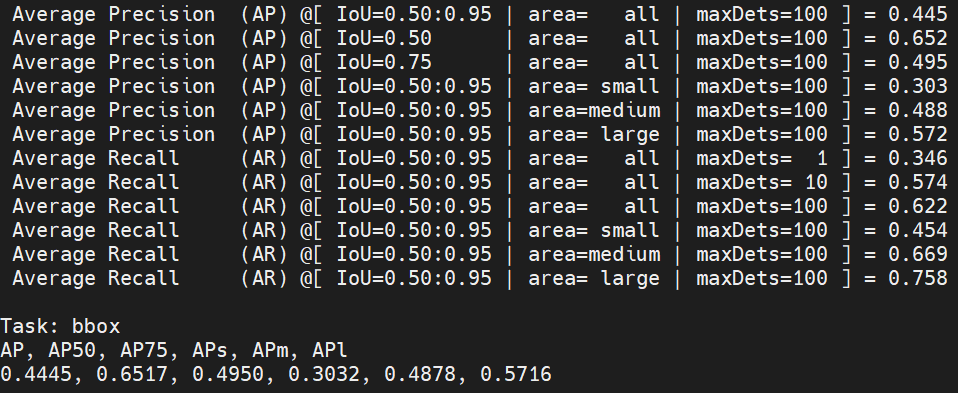
Yolov5m\_int8.bmodel



Yolov5l\_int8.bmodel

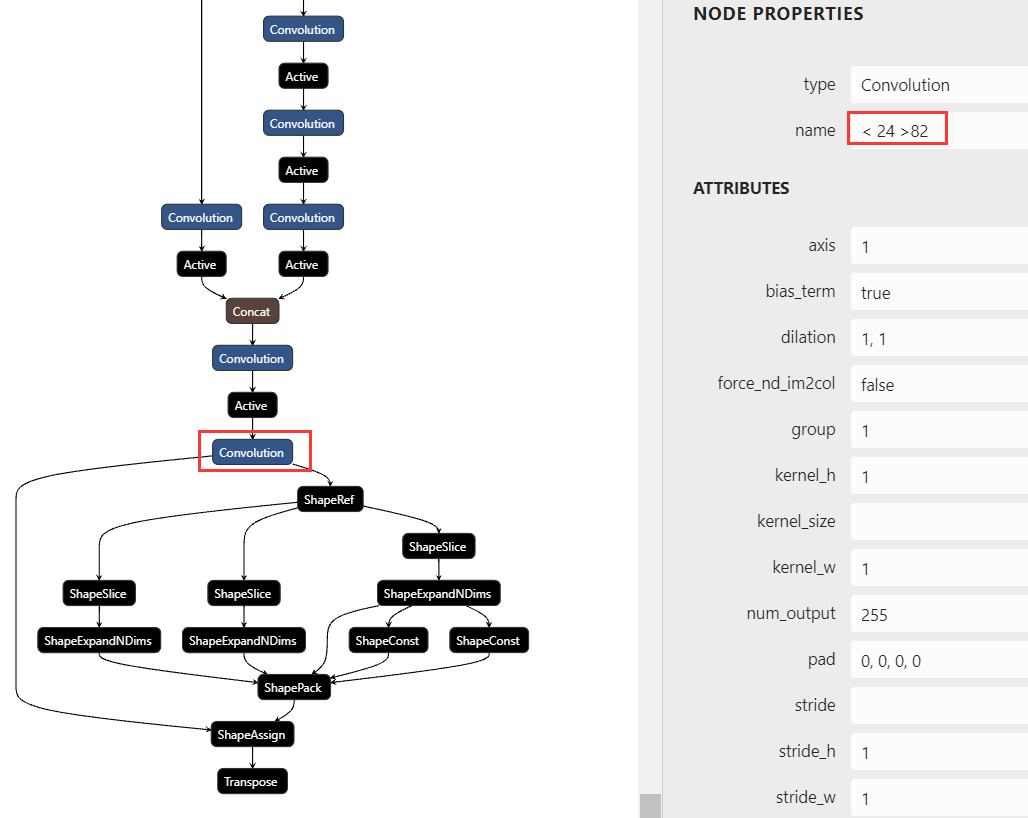


Yolov5x\_int8.bmodel

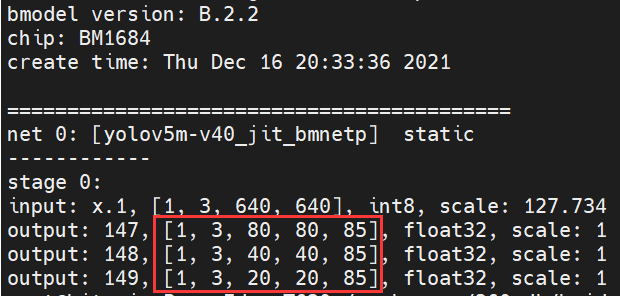


# Key point

Yololayer的输出是框跟得分，因为这两种数据分布不一样，所以无法量化，因此最后3个卷积之后选择fpfwd\_outputs用浮点计算。



# Output



对应3种不同尺度的网格(20,40,80)，每个网格有一个xywh的坐标一个class的得分(85)，因此每层输出是3\*80\*80\*85。

**对于一幅图像，分成20\*20的格子，每个格子预测2个边界框bounding box，而且每个边界框bounding box有x,y,w,h和confidence，5个参数，最后每个格子预测出属于C 个类别概率值**。

# autocalib一键量化

python3 ./scripts/calib\_model.py --model='/workspace/260sdk/baidu/yolov5m-v40\_jit.pt' --calib\_image\_source='/workspace/data/train2017' --data\_preprocess\_str='resize\_h=640,resize\_w=640;scale=0.0039,bgr2rgb=True' --input\_data\_shape='(1,3,640,640)' --calib\_iterations=10 --postprocess\_and\_calc\_score\_class=none --test\_calib\_opt\_accurcy='-fpfwd\_outputs="< 24 >14,< 24 >51,< 24 >82"'

注意，fpfwd\_outputs参数需要根据具体模型填写。**第一次量化如果结果错误一定要把log下的分布文件删除**。

# 训练

<https://blog.csdn.net/weixin_41010198/article/details/106785253>

训练选择在py37版本docker中进行

* 安装pytorch：

pip3 install torch==1.10.0+cpu torchvision==0.11.1+cpu torchaudio==0.10.0+cpu -f https://download.pytorch.org/whl/cpu/torch\_stable.html

* 可能会遇到的问题：tensorboard版本问题

pip3 install tensorboard==1.15.0

遇到setuptools distutils没有version的问题

修改/usr/local/lib/python3.7/site-packages/torch/utils/tensorboard/\_\_init\_\_.py

添加from distutils import version

pip3 install seaborn

* 训练

python3 train.py –img 640 –batch 16 –epochs 5 –data ./data/coco128.yaml –cfg ./models/yolov5m.yaml –weights yolov5m.pt

训练过程中会显示mAP，此结果可以忽略

* 跑测

python3 test.py --save-json –weights runs/exp2/weights/best.pt

